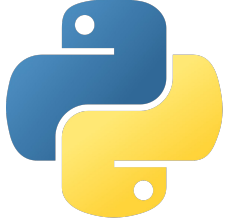
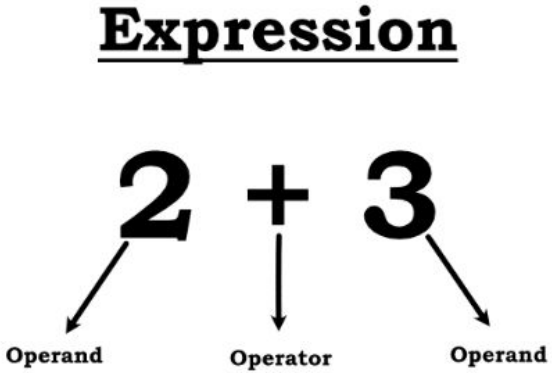


Chapter 2

Operators, Variables and Keywords

Operators vs Operands vs Expressions



Operators are special symbols that perform specific operations on one or more operands.

Operands are the values that an operator acts on.

A sequence (or combinations) of operands and operators, is called an expression

Operator Types

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. Logical Operators
5. Identity Operators

1. Arithmetic Operators

Operator	Name	Example
+	Addition	2 + 3
-	Subtraction	5 - 2
*	Multiplication	4 * 2
/	Division	10 / 2
%	Modulus	7 % 3
**	Exponentiation	2 ** 3
//	Floor division	10 // 3

2. Assignment Operators

Assignment operators are used to assign values to variables

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3

```
In [22]: x = 5

In [23]: print(x)
5

In [24]: x += 5

In [25]: print(x)
10

In [26]: x = x + 5

In [27]: print(x)
15
```

***Variables** are container, where we can store certain data value.*

3. Comparison Operators

Operator	Name	Example
==	Equal	7 == 7
!=	Not equal	10 != 3
>	Greater than	8 > 4
<	Less than	2 < 9
>=	Greater than or equal to	6 >= 6
<=	Less than or equal to	3 <= 2

Comparison operators are used to compare two values (operands or variables)

3. Comparison Operators (Lab)

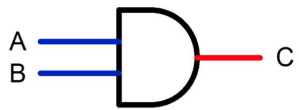
```
In [12]: 7 == 7
Out[12]: True

In [13]:

In [13]: 10 != 3
Out[13]: True

In [14]:

In [14]: 8 > 4
Out[14]: True
```

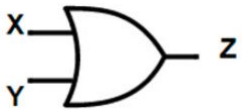
```
In [16]: 2 < 9
Out[16]: True

In [17]:

In [17]: 6 >= 6
Out[17]: True

In [18]:

In [18]: 3 <= 2
Out[18]: False
```

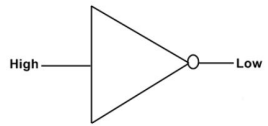
4. Logical Operators

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Logical operators are used to combine conditional expressions.







4. Logical Operators

```
In [21]: x = 5

In [22]:

In [22]: print(x > 3 and x < 10)
True

In [23]: print(x > 3 or x < 4)
True

In [24]: print(not(x > 3 and x < 10))
False
```

5. Identity Operators

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Identity operators are used to compare variables

5. Identity Operators (Lab)

```
In [2]: x = 5

In [3]: y = 10

In [4]:

In [4]: x is y
Out[4]: False

In [5]:

In [5]: x is not y
Out[5]: True
```

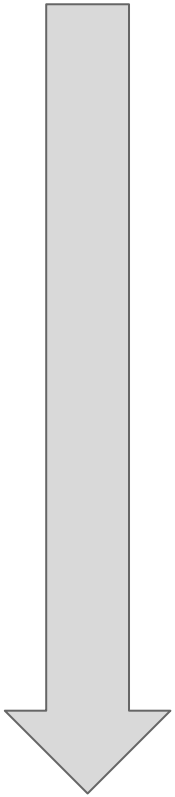
1

```
In [7]: x = 5

In [8]:

In [8]: y = x

In [9]:

In [9]: x is y
Out[9]: True

In [10]:

In [10]: x is not y
Out[10]: False
```

2

Operators Precedence

Precedence	Operators	Description	Associativity
1	() [], {}	Parentheses, Brackets, Braces	Left-to-right
2	**	Exponentiation	Right-to-left
3	+x, -x, ~x	Unary plus, Unary minus, Bitwise NOT	Right-to-left
4	*, /, //, %	Multiplication, Division, Modulo	Left-to-right
5	+, -	Addition, Subtraction	Left-to-right
6	<<, >>	Bitwise shift left, Bitwise shift right	Left-to-right
7	&	Bitwise AND	Left-to-right
8	^	Bitwise XOR	Left-to-right
9	|	Bitwise OR	Left-to-right
10	<, <=, >, >=	Comparison operators	Left-to-right
11	==, !=	Equality operators	Left-to-right
12	is, is not, in, not in	Identity and membership operators	Left-to-right
13	not	Logical NOT	Right-to-left
14	and	Logical AND	Left-to-right
15	or	Logical OR	Left-to-right
16	=, +=, -=, *=, /=, //=, %=, &=, ^=, |=, <<=, >>=	Assignment operators	Right-to-left



HIGHEST

LOWEST

Class Work

Q. Experiment below python expressions in interactive mode.

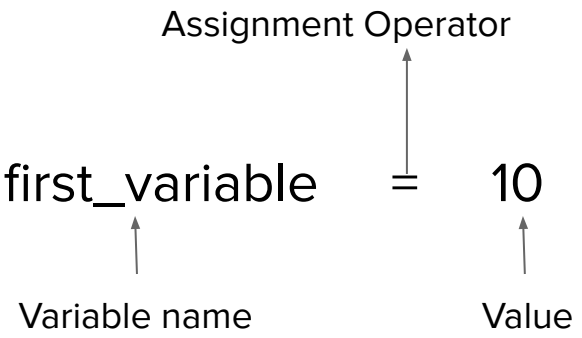
1. `(5 + 3) * 2 - 4 / 2`
2. `3 ** 2 * 4 + 5 // 2`
3. `10 / (2 + 3) * (8 - 4)`

Python Variables

Image Source

Variables

```
first_variable = 10
```



- **Variables are containers for storing data values.**
- When you assign a value to a variable, Python reserves a space in memory to store that value.

Rules for Creating Variables

1. **A variable name must start with a letter or the underscore character**
   - *True: ✅ (e.g., var_name, _variable)*
   - *False: ❌ (e.g., 123var, @variable)*

2. **A variable name cannot start with a number**
   - *True: ✅ (e.g., variable1, _variable2)*
   - *False: ❌ (e.g., 5number, 10th_variable)*

3. **A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)**
   - *True: ✅ (e.g., my_variable, var123)*
   - *False: ❌ (e.g., my_var$, var!able)*

4. **Variable names are case-sensitive (var, Var, VAR are three different variables)**

5. **A variable name cannot be any of the Python Keywords (*error*)**
   - *False: ❌ (e.g., for, if)*

Naming Conventions

1. **Camel Case:** Each word except the first, starts with a capital letter

```
camelCaseDemo = "camel case"
```

2. **Pascal Case:** Each word starts with a capital letter

```
PascalCaseDemo = "pascal case"
```

3. **Snake Case:** Each word is separated by underscore *(preferred in python)*

```
snake_case_demo = "snake case"
```

Assignments

1. Create a variable named variable test in snake case format and assign value “my first variable assignment”. Print the result.

2. Create a variable named age and assign value as 20. Print the result.

3. Create a variable called z, assign x + y to it, and Display the result.

   Initialize,
   x = 20, y = 30

4. Create three variables x, y, z and assign same value to all 3 variables in one code line.

5. Develop basic calculator app for the user:
   a. Input two numbers from user and assign it to variable num1 and num2
   b. Perform addition of two numbers and assign to variable result_add
   c. Display result_add to user.

Python Keywords and Identifiers

Python Keywords

- **Keywords** are the reserved words in Python.
- We cannot use keywords as variable name, function name, or any other identifier.
- Keywords are used to define the syntax and structure of the Python language.
- In Python keywords are case sensitive.
    - False is keyword but false is not.

```
if = 20
```

```
  File "<ipython-input-1-b8359eb0a5c6>", line 1
    if = 20
       ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

```
[2] for = 30
```

```
  File "<ipython-input-2-1f9609acbdb0>", line 1
    for = 30
        ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

Python 3.8 Keywords

```
In [5]: help('keywords')

Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not
```